Attorney Docket No.: 006570.P044 Utility Patent Application

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

FOR

MONITORING ARCHITECTURE EMPLOYING A HIERARCHICAL MONITOR TREE FOR MONITORING SYSTEM RESOURCES

INVENTORS:

GREGOR K. FREY
JOERG WELLER
JUERGEN OPGENORTH
REINHOLD KAUTZLEBEN
MIROSLAV R. PETROV

PREPARED BY:

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN
12400 Wilshire Boulevard, Seventh Floor
Los Angeles, California 90025-1026
(303) 740-1980

EXPRESS MAIL NO.

EV306655279US

MONITORING ARCHITECTURE EMPLOYING A HIERARCHICAL MONITOR TREE FOR MONITORING SYSTEM RESOURCES

BACKGROUND OF THE INVENTION

Field of the Invention

[0001] Embodiments of this invention generally relate to system resource monitoring and more particularly, to using a hierarchical monitor tree for monitoring of the resources.

Description of Related Art

[0002] In many modern computing systems and networks (systems), monitoring of system resources and components of significant importance to ensure not only the reliability and security of information flow, but also to promptly detect system deficiencies so that they are corrected in a timely manner. Typically, conventional client-server monitoring systems or architectures are used to perform monitoring of the resources.

[0003] **Figure 1** is a block diagram illustrating a conventional prior art monitoring system. As illustrated, typically, a conventional monitoring system (conventional system) 100 includes a monitor server 102 coupled with a client 104 and a data storage medium (storage) 110. The monitor server 102 is to monitor system resources (resources) 106-108 for the client 104. The monitor server 102 monitors the resources 106-108 in response to a request for monitoring received from the client 104. Typically, the storage 110 is used to store the monitoring data or information gathered by the monitoring server 102

from monitoring of the resources 106-108, and the monitoring data is then reported to the client 104. Stated differently, the data is not provided to the client 104 in real-time, delaying the detection and correction of deficiencies.

[0004] Conventional monitoring systems, however, do not provide real-time, continuous, or adequate monitoring of the resources, as they are limited in performance because of, for example, architectural limitations, inflexibility in levels of monitoring, lack of convenience in viewing the monitoring data, low level programming interface, and having the resources shared by all the software running on the system including virtual machines.

SUMMARY OF THE INVENTION

[0005] A method, apparatus, and system are provided for monitoring of system resources using a monitor tree. According to one embodiment, a resource may be associated with a monitor managed bean at a node of a monitor tree. Monitoring information regarding the associated resources may be requested from a runtime managed bean, and the monitoring information may be received by the monitor managed bean at the node of the monitor tree.

BRIEF DESCRIPTION OF THE DRAWINGS

[0006] The appended claims set forth the features of the present invention with particularity. The embodiments of the present invention, together with its advantages, may be best understood from the following detailed description taken in conjunction with the accompanying drawings of which:

[0007] **Figure 1** is a block diagram illustrating a conventional prior art monitoring system;

[0008] **Figure 2** is a block diagram illustrating an embodiment of a computer system;

[0009] **Figure 3** is a block diagram illustrating an embodiment of Java Management Extensions architecture;

[0010] **Figure 4** is a block diagram illustrating an embodiment of Java monitoring architecture;

[0011] **Figure 5** is a block diagram illustrating an embodiment of Java monitoring architecture;

[0012] **Figure 6** is a block diagram illustrating an embodiment of Java monitoring architecture;

[0013] **Figure 7** is a block diagram illustrating an embodiment of a Java monitoring architecture including a monitor tree;

[0014] **Figure 8** is a block diagram illustrating an embodiment of a tree node of a monitor tree;

[0015] **Figure 9A** is a block diagram illustrating an embodiment of a

monitoring data transmission sequence;

[0016] **Figur 9B** is a block diagram illustrating an embodiment of a monitoring data transmission sequence;

[0017] **Figure 9C** is a block diagram illustrating an embodiment of a monitoring data transmission sequence;

[0018] **Figure 10** is a flow diagram illustrating an embodiment of a process for monitoring of resources using a monitor tree;

[0019] **Figure 11** is a block diagram illustrating an embodiment of a node implementation;

[0020] **Figure 12A** is an exemplary illustration of a monitor tree using a monitor viewer according to one embodiment;

[0021] **Figure 12B** is an exemplary illustration of a monitor tree using a monitor viewer according to one embodiment;

[0022] **Figure 13A** is an exemplary illustration of a monitor tree using a monitor viewer according to one embodiment;

[0023] **Figure 13B** is an exemplary illustration of a monitor tree using a monitor viewer according to one embodiment;

[0024] **Figure 13C** is an exemplary illustration of a monitor tree using a monitor viewer according to one embodiment;

[0025] **Figure 14** is a block diagram illustrating an embodiment of an application server architecture; and

[0026] **Figure 15** is a block diagram illustrating an embodiment of an application server architecture.

DETAILED DESCRIPTION

[0027] A method, apparatus, and system are provided for monitoring of system resources using a monitor tree. Java objects, such as managed beans may be used for monitoring of the resources. The managed beans may include a combination of monitor managed beans (monitor beans) and runtime managed beans (runtime beans or resource beans). According to one embodiment, a monitor tree may be generated using instructions from a file from a central database. The monitor tree may include a number of nodes, each of the nodes having a monitor bean and a resource associated with the monitor bean. The monitor bean may request monitoring information from a runtime bean which may, in real-time, monitor one or more of the resources including the resource associated with the monitor tree. The runtime bean may provide, also in real-time, the monitoring information to the monitor bean at the node of the tree. The monitored resources may include Java resources associated with a Java 2 Platform, Enterprise Edition (J2EE) engine. Throughout this document, the use of the term "managed beans" in any combination may refer to Java Managed Beans (or Java MBeans). It should be noted, however, that the underlying principles of the invention are not limited to any particular application server specification or programming language.

[0028] According to one embodiment, runtime beans continuously provide monitoring information to monitor beans. According to another embodiment, the runtime beans provide the monitoring information to the monitor beans in

response to a request from the monitor beans. According to yet another embodiment, the runtime beans provide a notification signal (runtime notification) to the monitor beans indicating the availability of the monitoring information. The monitor bean may, in response to the runtime notification, request the monitoring information, which the runtime bean will provide. According to yet another embodiment, a timer may be employed to provide a notification signal (timer notification) to the monitor beans to request the monitoring information from the runtime beans. The timer notification may be based on, for example, predetermined time period or availability of the monitoring information.

[0029] According to one embodiment, a monitor service may be employed to connect the monitor tree with a central database and client-level applications. The client-level applications may include, for example, a computing center management system ("CCMS"), a well known management and monitoring architecture designed by SAP AG for use in R/3 systems. Additional and/or alternate destinations may include administrative tools/viewers, standard Web browsers, and third party tools/viewers. As described in detail below, one embodiment of the administration tools includes a monitor viewer for displaying the monitoring information (e.g., based on the structure of the monitoring tree). Client-level applications may also be used to originate and send request to the monitor tree to request monitoring information regarding various monitored resources.

[0030] In the following description, numerous specific details such as logic implementations, opcodes, resource partitioning, resource sharing, and resource

duplication implementations, types and interrelationships of system components, and logic partitioning/integration choices may be set forth in order to provide a more thorough understanding of various embodiments of the present invention. It will be appreciated, however, to one skilled in the art that the embodiments of the present invention may be practiced without such specific details, based on the disclosure provided. In other instances, control structures, gate level circuits and full software instruction sequences have not been shown in detail in order not to obscure the invention. Those of ordinary skill in the art, with the included descriptions, will be able to implement appropriate functionality without undue experimentation.

[0031] Various embodiments of the present invention will be described below. The various embodiments may be performed by hardware components or may be embodied in machine-executable instructions, which may be used to cause a general-purpose or special-purpose processor or a machine or logic circuits programmed with the instructions to perform the various embodiments. Alternatively, the various embodiments may be performed by a combination of hardware and software.

[0032] Various embodiments of the present invention may be provided as a computer program product, which may include a machine-readable medium having stored thereon instructions, which may be used to program a computer (or other electronic devices) to perform a process according to various embodiments of the present invention. The machine-readable medium may include, but is not limited to, floppy diskette, optical disk, compact disk-read-only

memory ("CD-ROM"), magneto-optical disk, read-only memory ("ROM") random access memory ("RAM"), erasable programmable read-only memory ("EPROM"), electrically erasable programmable read-only memory ("EEPROM"), magnetic or optical card, flash memory, or another type of media/machine-readable medium suitable for storing electronic instructions. Moreover, various embodiments of the present invention may also be downloaded as a computer program product, wherein the program may be transferred from a remote computer to a requesting computer by way of data signals embodied in a carrier wave or other propagation medium via a communication link (e.g., a modem or network connection).

[0033] **Figure 2** is a block diagram illustrating an embodiment of a computer system. The computer system (system) includes one or more processors 202-206 including one or more microprocessors, microcontrollers, field programmable gate arrays (FPGA), application specific integrated circuits (ASIC), central processing units (CPU), programmable logic devices (PLD), and similar devices that access instructions from system storage (e.g., main memory 216), decode them, and execute those instructions by performing arithmetic and logical operations.

[0034] The processor bus 212, also known as the host bus or the front side bus, may be used to couple the processors 202-206 with the system interface 214. The processor bus 212 may include a control bus 232, an address bus 234, and a data bus 236. The control bus 232, the address bus 234, and the data bus 236 may be multidrop bi-directional buses, e.g., connected to three or more bus agents, as opposed to a point-to-point bus,

which may be connected only between two bus agents.

[0035] The system interface 214 may be connected to the processor bus 212 to interface other components of the system 200 with the processor bus 212. For example, system interface 214 may includes a memory controller 218 for interfacing a main memory 216 with the processor bus 212. The main memory 216 typically includes one or more memory cards and a control circuit (not shown). System interface 214 may also include an input/output (I/O) interface 220 to interface the I/O bridge 224 with the processor bus 212. The I/O bridge 224 may operate as a bus bridge to interface between the system interface 214 and an I/O bus 226. One or more I/O controllers 228 and I/O devices 230 may be connected with the I/O bus 226, as illustrated. I/O bus 226 may include a peripheral component interconnect (PCI) bus or other type of I/O bus.

[0036] The system 200 may include a dynamic storage device, referred to as main memory 216, or a random access memory (RAM) or other devices coupled to the processor bus 212 for storing information and instructions to be executed by the processors 202-206. The main memory 216 also may be used for storing temporary variables or other intermediate information during execution of instructions by the processors 202-206. System 200 may include a read only memory (ROM) and/or other static storage device coupled to the processor bus 212 for storing static information and instructions for the processors 202-206.

[0037] The main memory 216 may include a wide variety of memory devices including read-only memory (ROM), erasable programmable read-only

memory (EPROM), electrically erasable programmable read-only memory (EEPROM), random access memory (RAM), non-volatile random access memory (NVRAM), cache memory, flash memory, and other memory devices. The main memory 216 may also include one or more hard disks, floppy disks, ZIP disks, compact disks (e.g., CD-ROM), digital versatile/video disks (DVD), magnetic random access memory (MRAM) devices, and other system-readable media that store instructions and/or data. The main memory 216 may store program modules such as routines, programs, objects, images, data structures, program data, and other program modules that perform particular tasks or implement particular abstract data types that facilitate system use.

[0038] The I/O device 230 may include a display device (not shown), such as a cathode ray tube (CRT) or liquid crystal display (LCD), for displaying information to an end user. For example, graphical and/or textual indications of installation status, time remaining in the trial period, and other information may be presented to the prospective purchaser on the display device. The I/O device 230 may also include an input device (not shown), such as an alphanumeric input device, including alphanumeric and other keys for communicating information and/or command selections to the processors 202-206. Another type of user input device includes cursor control, such as a mouse, a trackball, or cursor direction keys for communicating direction information and command selections to the processors 202-206 and for controlling cursor movement on the display device.

[0039] The system 200 may also include a communication device (not

shown), such as a modem, a network interface card, or other well-known interface devices, such as those used for coupling to Ethernet, token ring, or other types of physical attachment for purposes of providing a communication link to support a local or wide area network, for example. Stated differently, the system 200 may be coupled with a number of clients and/or servers via a conventional network infrastructure, such as a company's Intranet and/or the Internet, for example.

[0040] It is appreciated that a lesser or more equipped computer system than the example described above may be desirable for certain implementations. Therefore, the configuration of system 200 will vary from implementation to implementation depending upon numerous factors, such as price constraints, performance requirements, technological improvements, and/or other circumstances.

[0041] It should be noted that, while the embodiments described herein may be performed under the control of a programmed processor, such as processors 202-206, in alternative embodiments, the embodiments may be fully or partially implemented by any programmable or hardcoded logic, such as field programmable gate arrays ("FPGAs"), TTL logic, or application specific integrated circuits ("ASICs"). Additionally, the embodiments of the present invention may be performed by any combination of programmed general-purpose computer components and/or custom hardware components. Therefore, nothing disclosed herein should be construed as limiting the various embodiments of the present invention to a particular embodiment wherein the

recited embodiments may be performed by a specific combination of hardware components.

[0042] **Figure 3** is a block diagram illustrating an embodiment of Java Management Extensions architecture. The illustrated embodiment of Java Management Extensions ("JMX") architecture 300 includes three layers or levels 310, 320, 330. According to one embodiment, the three levels 310, 320, 330 may include a distributed services level (or manager or user or client level) 310, an agent level (or application level) 320, and an instrumentation level (or database level) 330. Some or all of the elements at each of levels of the JMX architecture 300 may be, directly or indirectly, interconnected via a network (e.g., a Local Area Network ("LAN")). Alternative embodiments of the JMX architecture 300 may include more or fewer levels.

[0043] The distributed services level 310 may serve as an interface between the JMX architecture 300 and one or more users or clients. As illustrated, the distributed services level 310 may include one or more user terminals 312-314. One or more of the user terminals 312-314 to collect and gather user input and send it to the agent level 320 over a network connection. Network connection may be a wired or wireless connection to a Local Area Network (LAN), a Wide Area Network (WAN), a Metropolitan Area Network (MAN), an intranet, and/or the Internet. Distributed services level terminals 312-314 may include personal computers, notebook computers, personal digital assistants, telephones, and the like. According to one embodiment in which network connection may connect to the Internet, one or more of the user

terminals 312-314 may include a Web browser (e.g., Internet Explorer or Netscape Navigator) to interface with the Internet.

[0044] According to one embodiment, the distributed services level 310 may also include management applications 316, such as a JMX-compliant management application, a JMX manager, and/or a proprietary management application. The management applications 316 may also include one or more graphical management applications, such as a visual administrator, operating to, for example, retrieve and display information received from the agent level 320 and/or the instrumentation level 330.

[0045] The visual administrator may include a monitor viewer to display such information. Management applications 316 may also include third party tools including a file system to store the information and may also include a Graphical User Interface ("GUI")-based monitor viewer to display the information. The distributed services level 310 may also include the CCMS system described above.

[0046] The agent level 320 may include one or more application servers 322-326. An application server may refer to a computing device that performs data processing. The agent level 320 may also include a computing device (e.g., a dispatcher) to perform load balancing among application servers 322-326. According to one embodiment in which the agent level 320 exchanges information with the distributed services level 310 via the Internet, one or more of the application servers 322-326 may include a Web application server. According to one embodiment, the application servers 322-326 may be

implemented according to the Java 2 Enterprise Edition Specification v1.4, published on July 12, 2002 ("the J2EE Standard"). In one embodiment of the invention, the management techniques described herein are used to manage resources within a "cluster" of server nodes. An exemplary cluster architecture is described below with respect to **Figures 14-15**. However, the underlying principles of the invention are not limited to any particular application server architecture.

[0047] The applications servers 322-326 may include one or more dedicated Java Managed Bean ("MBean") servers having agent services. According to one embodiment, for and at each Java virtual machine ("JVM") with managed resources, there may be one or more agents operating at the agent level 320. The one or more agents may include the one or more MBean servers, agent services, a set of MBeans, one or more connectors, and/or one or more protocol adaptors. An MBean Server may include a registry for MBeans and act as a single entry point for calling MBeans in a uniform fashion from management applications at other JVMs.

[0048] The instrumentation level 330 may provide a data storage medium for the JMX architecture 300. As illustrated, according to one embodiment, the instrumentation level 330 may include one or more database management systems ("DBMS") 332-334 and data sources 336-338. According to one embodiment, the data sources 336-338 may include databases and/or other systems capable of providing a data store.

[0049] Furthermore, according to one embodiment, the instrumentation

level 330 may include one or more hosts including one or more resources having MBeans, such as instrumentation MBeans. The instrumentation level 330 may make Java objects available to management applications 316. The Java objects instrumented according to the JMX-standard may include MBeans. According to one embodiment, the resources represented by MBeans may include managed resources 340. The managed resources 340 may include a kernel, a server component, or the like. MBeans may expose a management interface including constructors, attributes, operations, and notifications.

[0050] **Figure 4** illustrates one embodiment of the invention implemented within a JMX-based Java monitoring architecture (JMA) 400 for administration and management of the Java 2 Platform, Enterprise Edition ("J2EE") engine 406. According to one embodiment, the administration and management of the resources associated with the J2EE engine 406 may include monitoring of various system resources, including Java resources and other arbitrary resources and resources associated with the J2EE engine 406, including kernel, services, interfaces, libraries for each of the dispatchers and servers, network connections, memory consumption, threads, classloaders, database connections, database transactions, HyperText Transport Protocol ("HTTP") cache, Java Messaging Service ("JMS") queries and topics, sessions, and the like, using a monitoring service, such as the monitoring service 412.

[0051] According to one embodiment, various services of the monitoring service 412 may include monitoring of the resources, gathering of the monitoring data relating to the monitored resources, and maintaining of the monitoring data

410. The maintaining of the monitoring data 410 may include maintaining history and providing alerts when various resources, such as parameters, applications, or components reach a critical state, and such features may be enabled or disabled depending on when and whether the J2EE engine 406 is connected to the CCMS 422 via CCMS agent 402 with shared memory 404, directly or indirectly, coupled with the CCMS agent 402 and CCMS connector 408.

[0052] According to one embodiment, the JMA 400 may include the monitoring service 412 and one or more JMX-based monitor servers (JMX monitors). The monitoring service 412 may help establish a connection between a JMX monitor and the various components of the JMA 400. According to one embodiment, JMX monitors may reside and work on separate or remote Java virtual machines (JVMs) to collect data from cluster elements, and report information and statistics regarding the cluster nodes and their components to, for example, the visual administrator 414 having a monitor viewer 416, and/or to the CCMS 422 via the CCMS agent 402, and to various other third party tools. The CCMS 422, the visual administrator 414, and other third party tools may reside generally on the client side 420, while other components, as illustrated, may reside on the server side 418.

[0053] **Figure 5** illustrates an embodiment of the invention employed within a Java monitoring architecture ("JMA") 500. The illustrated embodiment includes a monitor server 502 comprised of a JMX-based monitors (JMX monitor) to, for example, monitor and collect data from various cluster elements and resources 510-514 of an application server engine. In one embodiment the

application server engine is a Java 2 Platform, Enterprise Edition (J2EE) engine (e.g., J2EE engine 406 of **Figure 4**). However, as stated above, the underlying principles of the invention are not tied to any particular specification or set of specifications. According to one embodiment, the monitor server 502 reports or transmit the collected data to various client-side 420 components and/or applications, such the visual administrator 414 having a monitor viewer 416. The monitor viewer 416 may be used to enable viewing of the monitoring data received from the monitor server 502.

[0054] The data collected may include information and statistics related to, for example, applications, server parameters, cluster nodes and their components of the J2EE engine. According to one embodiment, the collected data may also be reported to the CCMS 422 via a CCMS agent 402, and/or to the third party tools 518. In one embodiment, the third party tools 518 include a file system to, for example, temporarily store the collected data in a specified file format such as, for example, an Extensible Markup Language ("XML") format or a HyperText Markup Language ("HTML") format. The collected data may subsequently be reported to the components on the client-side 420 (e.g., in an XML or HTML format.

[0055] According to one embodiment, the expected overhead of the JMA 500 may vary according to its functionality. For example, the overhead may be light when reporting to the CCMS 422 or the third party tools 518, as opposed to when reporting to the visual administrator 414. Furthermore, the larger the requesting and/or reporting interval of the monitor server 502, the smaller the

expected overhead may be. The expected overhead may be relatively higher when using the monitor viewer 416 of the visual administrator 414 to actively retrieve and view the monitoring data.

[0056] **Figure 6** is a block diagram illustrating an embodiment of a Java monitoring architecture. According to one embodiment, as illustrated, the Java monitoring architecture (JMA) 600 may include a monitor service 602 to establish connection between one or more managed bean servers (bean servers) 604-608 with the rest of the JMA 600, including administrative tools having a visual administrator monitor viewer 610. The monitor viewer 610 may include a Graphical User Interface (GUI)-based monitor viewer or a monitor browser. The monitor service 602 may include a number of components including monitor servers and interfaces.

[0057] According to one embodiment, managed beans (MBeans or beans) (e.g., runtime managed beans or resources beans) 612-616 may be used to provide continuous monitoring of various resources 624-628 associated with a Java 2 Platform, Enterprise Edition (J2EE) engine, such as the J2EE 406 of **Figure 4**. The resources 624-628 include a kernel, services, interfaces, and libraries for dispatchers and servers, such as the dispatcher 618 and servers 620-622.

[0058] **Figure 7** is a block diagram illustrating an embodiment of a Java monitoring architecture including a monitor tree 714. Monitoring typically refers to a periodic oversight of a process, or the implementation of an activity, which seeks to establish the extent to which input deliveries, work schedules, required

actions and targeted outputs are proceeding according to plan, so that timely action can be taken to correct the deficiencies detected. According to one embodiment, Java monitoring system or architecture (JMA) 700 and its various components, including modules and servers, may be used to provide monitoring of resources and the semantics of data to ensure oversight, and may provide monitoring information to enable the proper analysis of the resources 720. Furthermore, according to one embodiment, JMA 600 may ensure first level of data processing, analysis, evaluating, and transporting of the data to various modules (e.g., monitor tree) and customer level components (e.g., CCMS 708, administrative tools, including a visual administrator 704, and third-party tools or plug-ins).

[0059] According to one embodiment, JMA 600 provides monitoring of the Java 2 Platform, Enterprise Edition (J2EE) engine 406 (see, e.g., **Figure 4**), and its associated resources 720, including but not limited to various managers, services, components, such as a kernel, interfaces, libraries for each of the dispatchers and servers, network connections, memory consumption, threads, classloaders, database connections, database transactions, HyperText Transport Protocol ("HTTP") cache, Java Messaging Service ("JMS") queues and topics, and sessions.

[0060] For monitoring of the resources 720, the JMA 700 may employ a monitor service 702 having modules, servers, and/or interfaces to connect the monitoring side with the rest of the JMA 700, including the central database 710 and client-side applications, such as the visual administrator 704 and CCMS

708. The monitoring architecture may include a managed bean server (bean server) 712, a plurality of monitor managed beans (monitor beans) 716 and/or runtime managed beans (runtime beans) 718. In one embodiment, the runtime beans 718 register with the bean server 712 to provide monitoring of the underlying resources 720 at runtime. The bean server 712 may include a container for the monitor beans 716 and the runtime beans 718, to provide them access to various resources 720 and management applications 704, 708.

[0061] To provide Java objects (e.g., monitor beans 716 and runtime beans 718) to various applications 704, 708 and to use the bean server 712 for such purposes, the applications 704, 708 may be made available via the distributed services level (e.g., distributed services level 310 of **Figure 3**) of the JMX-based architecture (e.g., JMA 700). For example, well-known protocols such as HTTP may be used to communicate with the bean server 712 via an appropriate protocol adapter that may translate operations into representations in a given protocol. In addition, one or more connectors may be used to communicatively couple the applications 704, 708 to the bean server 712 using a proprietary protocol.

[0062] The JMA 600 may be spread into three levels of the JMX architecture including a distributed services level 310, an agent level 320, and an instrumentation level 330 (see, e.g., **Figure 3**). The instrumentation level may include, for example, the monitor and runtime beans 716, 718, the agent level may include, for example, the bean server 712, and the distributed services level may include, for example, various applications 740, 708, adaptors, and

connectors.

[0063] According to one embodiment, various Java resources 720 at the agent level may be included as an external library. A JMX service module (jmx_service) may provide some of the functionality from the distributed services level, and may create an instance of the bean server 712 on each of the nodes in one or more clusters and provide local and cluster connections to all of them. The monitor beans 716 may be registered clusterwide. As such, a user or client may work with the bean server 712 transparently (e.g., from the user's perspective there may appear to be only one bean sever 712 showing all monitor beans 716 of the cluster). In addition, to receive JMX notifications clusterwide, a notification service module (jmx_notification) may be employed in combination with the JMX service,.

[0064] The bean server 712 may include a registry of the monitor and runtime beans 716-718, and the bean server 712 may serve as a single entry point for calling the monitor and runtime beans 716-718 in a uniform fashion from applications 704, 708 to monitor the resources 720, and collect or gather the monitoring data or information associated with the monitored resources 726. According to one embodiment, the bean server 712 may reside at a particular Java virtual machine (JVM) and may call the registered monitor and runtime beans 716-718 from the same JVM or other JVMs. According to one embodiment, the bean server 712 may not be limited to one bean server 712 and may include multiple managed bean servers. According to one embodiment, various modules of the monitor service 702 may also reside at the

same JVM along with the bean server 712, or may reside at other one or more JVMs that may be remotely located.

[0065] According to one embodiment, the resources 720 including kernel resources, libraries, interfaces, and services may be monitored using the runtime and monitor beans 716-718 registered with the bean server 712. To monitor the resources 720, including arbitrary Java resources, and be manageable, a management interface may be provided, and objects (e.g., monitor bean 716 and runtime bean 718) that implement such management interface may be registered with the bean server 712. The bean server 712, as previously described, may then serve as a container for the monitor bean 716 and the runtime bean 718, and provide them with access to the resources 720 to be monitored.

[0066] According to one embodiment, the runtime bean 718 (also sometimes referred to herein as a "resource" bean) may provide continuous or periodic monitoring of the resources 720 and may provide dynamic resource notifications or responses including information regarding the resources 720 to each of the monitor beans 716. According to one embodiment, the monitor service 702 may retrieve an Extensible Markup Language (XML) file 728, having semantics and installation directives on how a monitor tree 714 is created, from the database 710 to create the monitor tree 714. The monitor tree 714 may then be generated with various nodes, such as the node 730. Each of the nodes includes one or more monitor beans 716 associated with one or more resources 726.

[0067] According to one embodiment, the monitor bean 716 at the node 730 may request 722 information regarding its associated resource 726 from the runtime bean 718 associated with the resource. For example, the monitor bean 716 may invoke a method or request 722 during runtime to retrieve monitoring data from the runtime bean 718, and the runtime bean 718 may respond or provide a notification including the requested information 724 to the monitor bean 716 regarding the associated resource 726. According to another embodiment, the information 724 regarding the associated resource 726 may be provided periodically as predetermined and pre-set using a timer 732. The timer 732 may include predetermined criteria including a predetermined time period for periodically providing information 724 from the runtime bean 718 to the monitor bean 716.

[0068] The monitor service 702 may include an administrative services interface to provide the monitoring data or information to, for example, administrative tools including the visual administrator 704. The information received at the visual administrator 704 may be displayed using a monitor viewer 706 including a Web-based monitor browser or Graphical User Interface (GUI)-based monitor viewer. The monitor service 702 may include other interfaces, such as a managed enterprise Java beans ("MEJB") interface, to connect to remote third party tools, and a CCMS agent to connect to the CCMS 708.

[0069] The MEJB of the MEJB interface may include the following three types of beans: (1) session beans to perform processing; (2) entity beans to represent data, which may be a row, a column, or a table in a database, and (3)

message driven beans that are generated to process Java messaging service (JMS) messages. The MEJB may reside in and may be executed in a runtime environment (e.g., MEJB container), which may provide a number of common interfaces and service to the MEJB. The common interfaces and services may include security and transaction support. The MEJB interface may provide future scalability and allow multiple user interfaces to be used, such as a standard Web browser.

[0070] The managed beans, such as the monitor and runtime beans 716-718, may include the following two logical types of registered managed beans: (1) standard beans and (2) specific beans. Standard beans may provide standard functionality of start/stop, get/set properties, etc. Standard beans may be registered by default for all deployed components (e.g., kernel, libraries, interfaces, and services). Specific beans may provide component-specific functionalities that may vary from one component to another. To have the specific beans, a component may register an object that may implement a specific interface to list the processes available for its management and to extend the management interface (e.g., com.company.engine.frame.state. ManagementInterface).

[0071] According to one embodiment, for kernel resources, a standard bean may be registered with each manager having a specific bean. A prerequisite for this may be to return a non-null value in a method (e.g., getManagementInterface()) from the manager interface. For libraries and interfaces, only standard beans may be registered. For services, except for the

already registered standard beans, each of the services may register specific beans. Implementation of the management interface may also cause a specific bean to be registered for that particular service.

[0072] **Figure 8** illustrates an embodiment of a tree node 730 of a monitor tree. According to one embodiment, a hierarchical monitor tree 714 may be created to provide a grouping of monitoring agents (e.g., monitor bean 716) and the resources 726 associated with the monitoring agents, to provide a more manageable monitoring architecture. Although the monitoring agents and their corresponding resources may be grouped in a monitor tree, they are individually represented as tree nodes, and provide individual reporting of each of the resources, releasing the module developer from programmatically reporting the monitoring data to a central location.

[0073] For example, according to one embodiment, using the monitor tree, the module developer may configure runtime beans 718 to monitor one or more resources 720 and provide the monitoring information 724 to monitor beans 716 at a particular node 730 (or group of nodes). The monitoring information 724 provided to the monitor bean 716 may be provided continuously and/or upon request 722 from the monitor bean 716 associated with the underlying resource 726.

[0074] The associated resource 726 is associated with a particular monitor bean 716 to individualize the monitoring process (i.e., by receiving monitoring information about a particular resource, as opposed to about all of the resources 720). Similarly, according to one embodiment, particular resources 720 may be

associated with one or more runtime beans 718 to further individualize the process. The resources 720 may include the kernel, services, interfaces, libraries, and the like, for each of the dispatchers and servers associated with an application server engine (e.g., such as the Java 2 Platform, Enterprise Edition (J2EE) engine 406 described above).

[0075] As used herein, a "J2EE server" may include distributed J2EE servers, including, for example, multiple instances of a group of redundant J2EE servers and a dispatcher. The associated resource 726 may refer to one of the resources 720 that is associated with the monitor bean 716 at the node 730 to, for example, allow the monitor bean 716 to request monitoring information from the runtime bean 718. The end result is a simplified distribution of monitoring information in the form of a monitor tree with individual nodes, as opposed to a system in which monitoring information must be centralized.

[0076] According to one embodiment, the monitoring data 724 may then be provided to various client level applications, such as the visual administrator 704 and/or CCMS 708 via the monitor service 702. According to one embodiment, the monitor viewer 706 may be used to view the monitor tree (e.g., monitor tree 714 of **Figure 7**) and its various tree nodes and the monitoring information associated with the each of the nodes (e.g., node 730 and the associated resource 726). A user may select any of the nodes or resources shown as part of the monitor tree to view the monitoring data. According to one embodiment, a color or symbol combination may be used to determine the status of associated resources at various nodes. For example, a red/green/yellow

combination may be used wherein, if the associated resource 726 at the node 730 is in a "normal" state, a green mark may be shown next to it. If the associated resource 726 approaches a critical value, the mark may turn yellow and once the associated resource 726 reaches the critical value, the mark may turn red. Additionally, the mark may be of a different color, such as white, when there is inactivity associated with the associated resource 726. However, the underlying principles of the invention are not limited to any particular color scheme.

[0077] According to one embodiment, the monitor viewer 706 may be used to display other information related to the resources 720 including, for example, the name of the resource, the resource description, the resource type, relevant dates and/or time, resource properties and values. The information may also include data history regarding the resources 720. The data history may include, for example, values associated with the resource over different periods of time (e.g., over a minute, hour, day, . . . etc).

[0078] As mentioned above, according to one embodiment, an XML file 728 may be retrieved by the monitor service 702 from a central database 710 to generate a monitor tree 714. The XML file may include semantics and directives to generate the monitor tree using the monitor service 702. An XML parser may be used to parse the semantics and directives of the XML file. Furthermore, a Java Management Extensions (JMX)-based application programming interface (API) may be used to create and customize the monitor tree. The monitor API may facilitate and handle both (1) the creation of the monitor tree and (2) the

customization of the monitor tree as will as the reporting of the monitoring information and values. Typically, an API refers to a language or message format used by an application program to communicate with the operating system, communications protocols, and/or with other control programs (e.g., database management systems).

[0079] **Figure 9A** illustrates an embodiment of a monitoring data transmission sequence. As illustrated, a monitor managed bean 716 may receive monitoring information from a runtime managed bean 718. The monitor bean 716 may be associated with a particular resource 726 of the resources 720 being monitored. The runtime bean 718 may provide monitoring information 902 to the monitor bean 716 at a tree node 730 of the monitor tree 714. According to one embodiment, the runtime bean 718 may continuously provide such information 902 to the monitor bean 716, without receiving a request (or any other such indication or signal) from the monitor bean 716.

[0080] **Figure 9B** illustrates another embodiment of a monitoring data transmission sequence. As illustrated, in this embodiment, the runtime bean 718 may send a notification 904 to the monitor bean 716 related to the monitoring of the associated resource 726. For example, in one embodiment, the runtime bean 718 may transmit a notification upon detecting a specified event associated with its resource (e.g., such as the resource reaching a critical value). Upon receiving the notification 904 from the runtime bean 718, the monitor bean 716 may send a request 906 to the runtime bean 718, requesting the monitoring data. In response to the request 906, the runtime bean 718 may send the

information 902 including monitoring data regarding the associated resource 726 to the monitor bean 716.

[0081] **Figure 9C** illustrate another embodiment of a monitoring data transmission sequence. As illustrated, in this embodiment, a timer 732 is configured to manage transmission of the monitoring data. The timer 732 may be in communication with a managed bean server 712 to help facilitate a timetable (or schedule or pattern) for transmitting the monitoring data. For example, the timer 732 may include a predetermined period of time after which the timer will signal to the monitor bean 716 to place a query with the runtime bean 718. Upon receiving the notification 908, the monitor bean 716 may send the request 906 to the runtime bean 718, requesting the monitoring data. In response to the request 906, the runtime bean 718 may provide the information 902 including the monitoring data regarding the associated resource 726 to the monitor bean 716.

[0082] **Figure 10** is a flow diagram illustrating an embodiment of a process for monitoring of resources using a monitor tree. First, according to one embodiment, at processing block 1002, a resource (associated resource) of the monitoring resources may be associated with a monitor managed bean at a node of a monitor tree so that monitoring information regarding the associated resource may be retrieved from runtime managed bean. The runtime managed bean may monitor one or more of the resources including Java resources associated with a Java 2 Platform, Enterprise Edition (J2EE) engine. As mentioned above, the Java resources may include, for example, kernel

resources, services, interfaces, and libraries for each of the dispatcher and server associated with the J2EE engine. The Java resources may also include network connections, memory consumption, threads, classloaders, database connections, database transactions, HyperText Transport Protocol (HTTP) cache, Java Messaging Service (JMS) queries and topics, sessions, and the like.

[0083] At decision block 1004, it is determined whether timer notification service is being utilized. If the timer notification service is employed, the timer notification may be received at the monitor bean from the timer at processing block 1006. The request for the monitoring information regarding the associated resource may then be placed by the monitor bean with the runtime bean at processing block 1012. However, if the notification service is not employed, the process may continue with decision block 1008.

[0084] At decision block 1008, a determination is made as to whether a runtime bean notification service is used. As described above, if the runtime bean notification service is used, the runtime bean may provide a notification to the monitor bean in response to a specified event associated with the resource (e.g., when the monitored value reaches a threshold). Upon receiving the runtime notification, at processing block 1010, the monitor bean may request the monitoring data associated with the associated resource at processing block 1012.

[0085] With neither the timer notification service nor the runtime notification service is employed, the monitor bean may place a request with the runtime bean seeking the monitoring information regarding the associated

resource at processing block 1012. The monitor bean may then receive such monitoring information from the runtime bean at processing block 1014. The process may then continue with processing block 1004.

[0086] **Figure 11** is a block diagram illustrating an embodiment of a node implementation. According to one embodiment, the node 1100 may include one or more processors 1102 (e.g., processors 202-206 of Figure 2), one or more memory devices 1104 (e.g., main memory 216 of Figure 2), one or more Input/Output (I/O) devices 1106 (e.g., I/O devices 230 of Figure 2), one or more network interfaces 1108, and Java monitoring system or architecture (JMA) 1110 (e.g., JMA 700 of Figure 7), directly or indirectly, connected together through a system interconnect or network 1112. The processors 1102 may include microprocessors, microcontrollers, field programmable gate arrays (FPGAs), application specific integrated circuits (ASICs), central processing units (CPUs), programmable logic devices (PLDs), and similar devices that access instructions from a system storage (e.g., memory 1104), decode them, and execute those instructions by performing arithmetic and logical operations.

[0087] The JMA 1110 may include a monitor tree (e.g., monitor tree 714 of Figure 7) used for monitoring of resources including Java resources associated with, for example, a Java 2 Platform, Enterprise Edition (J2EE) engine. Components for monitoring of the Java resources using the JMA 1110 may include executable content, control logic (e.g., ASIC, PLD, FPGA, etc.), firmware, or some combination thereof, in one embodiment of the present invention. In embodiments of the invention in which Java monitoring may include executable

content, it may be stored in the memory device 1104 and executed by the control processor 1102.

[0088] Memory devices 1104 may encompass a wide variety of memory devices including read-only memory (ROM), erasable programmable read-only memory (EPROM), electrically erasable programmable read-only memory (EEPROM), random access memory (RAM), non-volatile random access memory (NVRAM), cache memory, flash memory, and other memory devices. Memory devices 1104 may also include one or more hard disks, floppy disks, ZIP disks, compact disks (e.g., CD-ROM), digital versatile/video disks (DVD), magnetic random access memory (MRAM) devices, and other system-readable media that store instructions and/or data. Memory devices 1104 may store program modules, such as routines, programs, objects, images, data structures, program data, and other program modules that perform particular tasks or implement particular abstract data types that facilitate system use.

[0089] The I/O devices 1106 may include hard disk drive interfaces, magnetic disk drive interfaces, optical drive interfaces, parallel ports, serial controllers or super I/O controllers, serial ports, universal serial bus (USB) ports, display device interfaces (e.g., video adapters), network interface cards (NICs), sound cards, modems, and the like. System interconnect or network 1112 may permit communication between the various elements of node 1100. System interconnects 1112 may include a wide variety of signal lines including one or more of memory buses, peripheral buses, local buses, host buses, and bridge, optical, electrical, acoustical, and other propagated signal lines.

[0090] **Figure 12A** is an exemplary illustration of a monitor tree as displayed within a monitor viewer, according to one embodiment. A monitor tree 1202 is illustrated having several nodes 1204. Each of the nodes 1204 is associated with a corresponding monitoring resource 1206 (e.g., applications, kernel, services, and system).

[0091] **Figure 12B** is another view from within the monitor viewer according to one embodiment. As with **Figure 12A**, a monitor tree 1202 is illustrated having several nodes 1204. Each of the nodes 1204 is associated with a corresponding monitoring resource 1206 (e.g., applications, kernel, services, and system). Further, various colors marks 1208-1214 representing corresponding status of the resources 1206 are illustrated here as different symbols.

[0092] For example, the green mark 1208, represented here as a circle, may indicate monitoring of the corresponding resource. The yellow mark 1210, represented here as a triangle, may indicate continuous monitoring of the corresponding resource and/or may indicate that the resource being monitored is close to a critical value or stage. The red mark 1212, represented here as a square, may indicate that the corresponding resource has reached a critical value. Finally, the white mark 1214, represented here as a diamond, may indicate inactivity or that the corresponding resource is not being monitored.

[0093] **Figure 13A** is another exemplary illustration a monitor viewer according to one embodiment. As described with reference to **Figures 12A-12C**, the monitor tree 1302 includes various tree nodes and their associated

resources, such as the kernel 1334. According to one embodiment, the monitor viewer may be used to view general information 1304 relating to any of the resources and their subcategories. For example, the general information 1304 regarding the Kernel/ClassLoader Manager/ClassLoadersCount 1334, 1336, 1306 may include the name 1308, description 1310, type 1312, creation date 1314, last change date 1316, and other monitoring data 1318. The selection under general information 1304 may also include information relating to monitor configuration 1320 and monitor history 1322.

[0094] **Figure 13B** is another exemplary illustration of a monitor viewer according to one embodiment. As described with reference to **Figure 13A**, the general information 1304 may include information relating to monitor configuration 1320 as it relates to the properties of Kernel/ClassLoader Manager/ClassLoadersCount 1334, 1336, 1306. The monitor configuration 1320 may include various color changing thresholds 1324 including, for example, changes from green to yellow 1334, changes from yellow to red 1336, changes from red to yellow 1338, and changes from yellow from green 1340.

[0095] **Figure 13C** is another exemplary illustration of a monitor viewer according to one embodiment. As described with reference to **Figure 13A**, the general information 1304 may include information relating to monitor history 1322 as it relates to the properties of Kernel/ClassLoader Manager/ ClassLoadersCount 1334, 1336, 1306. The monitor history 1322 may include monitoring data history relating to ClassLoadersCount 1322 for various time periods 1326-1332. The various time periods 1326-1332 may include the minute

history 1326, the five-minute history 1328, the quarter or fifteen-minute history 1330, and the hour history 1332. It is contemplated that the monitor history 1322 may include other time periods, such as a thirty-minute history, a two-hour history, and the like.

[0096] In one embodiment of the invention, the management techniques which are the focus of this application are used to manage resources within a cluster of server nodes. An exemplary application server architecture will now be described, followed by a detailed description of the management architecture and associated processes.

[0097] An application server architecture employed in one embodiment of the invention is illustrated in **Figure 14**. The architecture includes a central services "instance" 1400 and a plurality of application server "instances" 1410, 1420. As used herein, the application server instances, 1410 and 1420, each include a group of server nodes 1414, 1416, 1418 and 1424, 1426, 1428, respectively, and a dispatcher, 1412, 1422, respectively. The central services instance 1400 includes a locking service 1402 and a messaging service 1404 (described below). The combination of all of the application instances 1410, 1420 and the central services instance 1400 is referred to herein as a "cluster." Although the following description will focus solely on instance 1410 for the purpose of explanation, the same principles apply to other instances such as instance 1420.

[0098] The server nodes 1414, 1416, 1418 within instance 1410 provide the business and/or presentation logic for the network applications supported by

the system. Each of the server nodes 1414, 1416, 1418 within a particular instance 1410 may be configured with a redundant set of application logic and associated data. In one embodiment, the dispatcher 1412 distributes service requests from clients to one or more of the server nodes 1414, 1416, 1418 based on the load on each of the servers. For example, in one embodiment, a dispatcher implements a round-robin policy of distributing service requests (although various alternate load balancing techniques may be employed).

[0099] In one embodiment of the invention, the server nodes 1414, 1416, 1418 are Java 2 Platform, Enterprise Edition ("J2EE") server nodes which support Enterprise Java Bean ("EJB") components and EJB containers (at the business layer) and Servlets and Java Server Pages ("JSP") (at the presentation layer). Of course, certain aspects of the invention described herein may be implemented in the context of other software platforms including, by way of example, Microsoft .NET platforms and/or the Advanced Business Application Programming ("ABAP") platforms developed by SAP AG, the assignee of the present application.

[0100] In one embodiment, communication and synchronization between each of the instances 1410, 1420 is enabled via the central services instance 1400. As illustrated in **Figure 14**, the central services instance 1400 includes a messaging service 1404 and a locking service 1402. The message service 1404 allows each of the servers within each of the instances to communicate with one another via a message passing protocol. For example, messages from one server may be broadcast to all other servers within the cluster via the messaging

service 1404. In addition, messages may be addressed directly to specific servers within the cluster (i.e., rather than being broadcast to all servers).

[0101] In one embodiment, the locking service 1402 disables access to (i.e., locks) certain specified portions of configuration data and/or program code stored within a central database 1430. A locking manager 1440, 1450 employed within the server nodes locks data on behalf of various system components which need to synchronize access to specific types of data and program code (e.g., such as the configuration managers 1444, 1454 illustrated in **Figure 14**). As described in detail below, in one embodiment, the locking service 1402 enables a distributed caching architecture for caching copies of server/dispatcher configuration data.

[0102] In one embodiment, the messaging service 1404 and the locking service 1402 are each implemented on dedicated servers. However, the messaging service 1404 and the locking service 1402 may be implemented on a single server or across multiple servers while still complying with the underlying principles of the invention.

[0103] As illustrated in **Figure 14**, each server node (e.g., 1418, 1428) includes a lock manager 1440, 1450 for communicating with the locking service 1402; a cluster manager 1442, 1452 for communicating with the messaging service 1404; and a configuration manager 1444, 1454 for communicating with a central database 1430 (e.g., to store/retrieve configuration data). Although the lock manager 1440, 1450, cluster manager 1442, 1452 and configuration manager 1444, 1454 are illustrated with respect to particular server nodes, 1418

and 1428, in **Figur 14**, each of the server nodes 1414, 1416, 1424 and 1426 and/or on the dispatchers 1412, 1422 may be equipped with equivalent lock managers, cluster managers and configuration managers.

[0104] Referring now to **Figure 15**, in one embodiment, configuration data 1520 defining the configuration of the central services instance 1400 and/or the server nodes and dispatchers within instances 1410 and 1420, is stored within the central database 1430. By way of example, the configuration data may include an indication of the kernel, applications and libraries required by each dispatcher and server; network information related to each dispatcher and server (e.g., address/port number); an indication of the binaries required during the boot process for each dispatcher and server, parameters defining the software and/or hardware configuration of each dispatcher and server (e.g., defining cache size, memory allocation, . . . etc); information related to the network management configuration for each server/dispatcher (e.g., as described below); and various other types of information related to the cluster.

[0105] In one embodiment of the invention, to improve the speed at which the servers and dispatchers access the configuration data, the configuration managers 1444, 1454 cache configuration data locally within configuration caches 1500, 1501. As such, to ensure that the configuration data within the configuration caches 1500, 1501 remains up-to-date, the configuration managers 1444, 1454 may implement cache synchronization policies.

[0106] It should be appreciated that reference throughout this specification to "one embodiment" or "an embodiment" means that a particular feature,

structure or characteristic described in connection with the embodiment is included in at least one embodiment of the present invention. Therefore, it is emphasized and should be appreciated that two or more references to "an embodiment" or "one embodiment" or "an alternative embodiment" in various portions of this specification are not necessarily all referring to the same embodiment. Furthermore, the particular features, structures or characteristics may be combined as suitable in one or more embodiments of the invention.

[0107] Similarly, it should be appreciated that in the foregoing description of exemplary embodiments of the invention, various features of the invention are sometimes grouped together in a single embodiment, figure, or description thereof for the purpose of streamlining the disclosure aiding in the understanding of one or more of the various inventive aspects. This method of disclosure, however, is not to be interpreted as reflecting an intention that the claimed invention requires more features than are expressly recited in each claim. Rather, as the following claims reflect, inventive aspects lie in less than all features of a single foregoing disclosed embodiment. Thus, the claims following the detailed description are hereby expressly incorporated into this detailed description, with each claim standing on its own as a separate embodiment of this invention.

[0108] While certain exemplary embodiments have been described and shown in the accompanying drawings, it is to be understood that such embodiments are merely illustrative of and not restrictive, and that the embodiments of the present invention are not to be limited to specific

constructions and arrangements shown and described, since various other modifications may occur to those ordinarily skilled in the art upon studying this disclosure.